

Combination of Keyword Extraction with Text Clustering by String Vector based KNN and AHC Algorithm

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the string vector based KNN algorithm and the AHC algorithm as the respective approaches to the keyword extraction and the text clustering. In the previous work, even if we proposed the string vector based KNN algorithm as an approach to the keyword extraction, a domain should be presented as a tag of an input text for carrying out the task. In this research, a text group is made into subgroups based on their contents, and keywords are extracted from a novice text which is separated from the group by a classifier which corresponds to its most similar cluster. The AHC algorithm and the KNN algorithm which process string vectors directly are adopted respectively as a text clustering tool and a keyword extraction tool. The goals of this research are to automate the keyword extraction and to improve the performance of both tasks, using two string vector-based algorithms.

I. INTRODUCTION

The keyword extraction is the process of showing important and relevant to the input text. The keyword extraction is viewed into a binary classification, to apply the machine learning algorithm. The process of keyword extraction is to index a text into words, classify each word into keyword or non-keyword, and extract ones which are classified with keyword. The binary classification which is mapped from the keyword extraction belongs to the domain dependent classification where a same word may be classified differently depending on the domain. It is necessary to present the domain in addition to the input text for performing the keyword extraction.

This research is motivated by the necessity of defining the domains for designing the keyword extraction system. A keyword extraction system is composed with binary classification tasks each of which corresponds to a domain. The process of designing the keyword extraction system is to predefine the domains, collect sample examples domain by domain, and allocate a classifier for each domain. The results from applying the string vector based KNN algorithm and the string vector based AHC algorithm respectively to the keyword extraction and the text clustering were successful. This research is intended to automate the domain predefinition by connecting the keyword extraction with the text clustering.

The idea of this research is to apply the string vector based KNN algorithm and the string vector based AHC algorithm to the keyword extraction which relates to the text clustering. In this research, it is assumed that the collection of texts each of which is initially associated with its own keywords is prepared. The KNN algorithm and the AHC algorithm are modified into the string vector-based versions as the approaches to the keyword extraction and the text clustering. The text collection is segmented into clusters each of which corresponds to a domain, and a binary classifier is allocated to each cluster. A domain is automatically decided to an input text by its similarities with the cluster prototypes, to select a corresponding classifier.

Let us mention some benefits from this research. The main problems in encoding words and texts into numerical vectors are avoided by encoding them into string vectors. The performances in the keyword extraction and the text clustering are improved by adopting the versions of the KNN algorithm and the AHC algorithm which process directly string vectors. The domains are automatically predefined through the text clustering; prior knowledge or subjective views are not required for defining the domains. A domain is automatically decided by the similarities of an input text with clustering prototypes; it is not required as an input text tag.

Let us mention some remaining tasks as the further discussions on this research. We need to modify more advanced machine learning algorithms and deep learning algorithms into the string vector-based versions. It is necessary to define and characterize mathematically other operations on string vectors for doing that. The keyword extraction system which relates to the text clustering should be implemented as a real program or a library. The keyword extraction may be applied for reinforcing the performances of main text mining tasks, such as the text classification and the text clustering.

II. PROPOSED APPROACH

A. Similarity Metric

This section is concerned with the string vector as the representation of a word and the similarity between string vectors. The string vector is defined as a finite ordered set

of strings; in the vector, the numerical values are replaced by the strings. It is required to define a similarity matrix for computing the similarity between elements. The similarity between string vectors is the average over one-to-one similarities between their elements. This section is intended to describe the process of encoding a word into a string vector and the process of computing the similarity between string vectors.

The process of encoding words into string vectors is illustrated in Figure 1. In the word-text association, a text collection is constructed by gathering texts, and each word is associated with its own texts based on their information. The string vector features are defined as symbolic forms manually in the feature definition. In the feature value assignment, the string vector which represents a word is filled with text identifiers which correspond to the features. Refer to [1] for studying the encoding process in more detail.

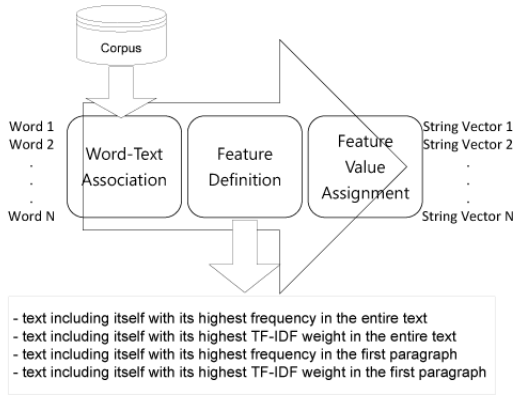


Figure 1. Process of Encoding Words into String Vectors

The similarity matrix which is the basis for computing the similarity between string vectors is illustrated in figure 00. In the matrix, each row and each column correspond to text identifiers, and each element is the similarity between texts, d_i and d_j , as shown in equation (1),

$$s_{ij} = \text{sim}(d_i, d_j) \quad (1)$$

The similarity between two texts, d_i and d_j , is computed by equation (2),

$$\text{sim}(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|} \quad (2)$$

where D_i is the set of words in the text, d_i , and D_j is the set of words in the text, d_j . The value of similarity between texts, d_i and d_j , is always given as a normalized value between zero and one, as shown in equation (3)

$$0 \leq \text{sim}(d_i, d_j) \leq 1 \quad (3)$$

The similarity matrix is used for computing the similarity between string vectors which represent words as a reference table.

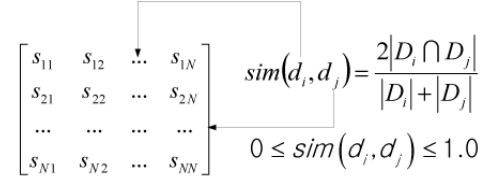


Figure 2. Semantic Similarity Matrix

Let us mention the process of computing the similarity between string vectors as a semantic similarity between words. The two string vectors which represent words are notated by $\text{str}_1 = [d_{11} \ d_{12} \ \dots \ d_{1d}]$ and $\text{str}_2 = [d_{21} \ d_{22} \ \dots \ d_{2d}]$. The similarity between two string vectors is computed by equation (4),

$$\text{sim}(\text{str}_1, \text{str}_2) = \frac{1}{d} \sum_{i=1}^d \text{sim}(d_{1i}, d_{2i}). \quad (4)$$

The similarity between elements, $\text{sim}(d_{1i}, d_{2i})$, is taken from the similarity which was mentioned above. The value which is generated by equation (4) is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(\text{str}_1, \text{str}_2) \leq 1. \quad (5)$$

Let us make some remarks on the process of encoding words into string vectors and computing the similarity between words. A word is encoded into a string vector by the word-text association, the feature definition, and the feature value assignment. The similarity matrix is defined as the basis for computing the similarity between string vectors. The similarity is computed by equation (4). In the future research, we consider representing a word into multiple string vectors.

B. String Vector based KNN Algorithm

This section is concerned with the string vector based KNN algorithm. It was initially proposed as the approach to the text summarization by Jo in 2018 [1]. The similarity metric between string vectors which was described in the previous section is used for computing the similarity between a novice string vector and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 00. The labeled words which are gathered as samples are encoded into string vectors, and they are notated by a set, $Tr = \{\text{str}_1, \text{str}_2, \dots, \text{str}_N\}$. A novice word is encoded into a string vector notated by str . Its similarities with the string vectors which represent the sample words are computed by equation (4), as

$sim(\mathbf{str}, \mathbf{str}_1), sim(\mathbf{str}, \mathbf{str}_2), \dots, sim(\mathbf{str}, \mathbf{str}_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

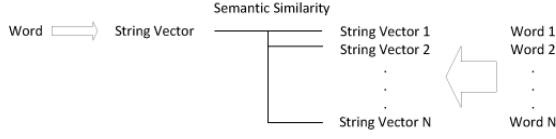


Figure 3. Similarities of Novice Input with Training Examples

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = Select_k(Tr, \mathbf{str}), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{\mathbf{str}_1^{near}, \mathbf{str}_2^{near}, \dots, \mathbf{str}_k^{near}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

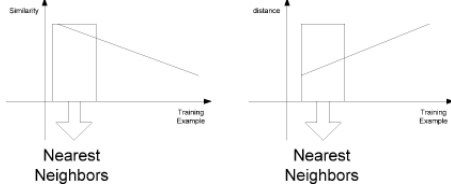


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = label(\mathbf{str}_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $Count(Nr, c_j)$. The label of the novice item, $\mathbf{str}$, is decided by equation (8),

$$label(\mathbf{str}) = \underset{j=1}{\operatorname{argmax}}^m Count(Nr, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.

Let us make some remarks on the string vector-based version of KNN algorithm. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors.



Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the keyword extraction system which adopts the string vector based KNN algorithm. In Section II-B, we described the proposed version of KNN algorithm as the approach to the keyword extraction. It is viewed as a binary classification which classifies a word into keyword or non-keyword. This section is intended to describe the keyword extraction system with respect to its function and its architecture.

The process of collecting the sample words and encoding them into string vectors for implementing the keyword extraction system is illustrated in Figure 6. The keyword extraction is viewed as a binary classification which classifies a word into keyword or non-keyword. The topic-based word classification belongs to the domain independent classification where a same word is always classified identically with regardless of the domain, whereas the keyword extraction is the domain dependent classification where a same word may be classified differently depending on the domain. The words which are labeled with keyword or non-keyword are collected domain by domain. It is required to tag the input text with its own domain for executing the keyword extraction.

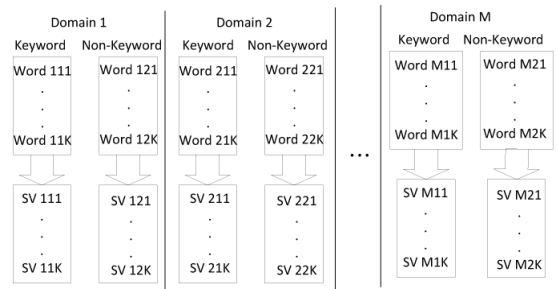


Figure 6. Preparation of Training Examples

The entire architecture of the proposed keyword extraction system is illustrated in Figure 7. A text is given as the input, and words are extracted from it in the indexing module. In the encoding module, sample words in the keyword group and the non-keyword group and ones which are indexed from the text are mapped into string vectors. The words which are indexed from the text are classified into one of the two categories in the similarity computation module and

the voting module. The system consists of the four modules: indexing module, encoding module, similarity computation module, and voting module.

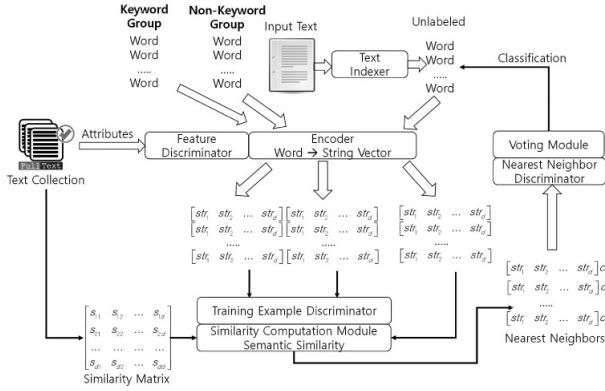


Figure 7. System Architecture

The execution process of the proposed system is illustrated as a block diagram in Figure 8. Sample words which are labeled with keyword or non-keyword are collected within a domain, and they are encoded into string vectors. An input text is indexed into a list of words, and they are also encoded into string vectors. The nearest neighbors are selected by the similarity computation, and the label is decided by voting ones of their nearest neighbors for each word. The words which are classified into keyword are extracted as the final output.

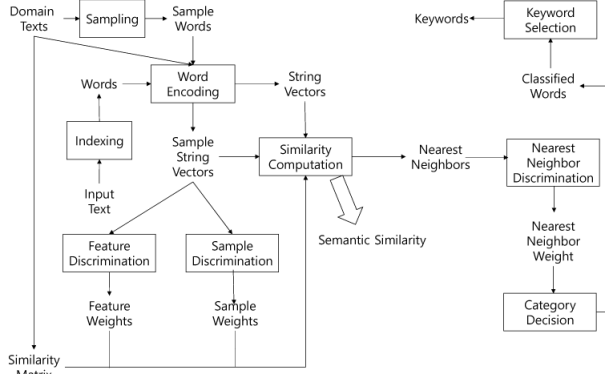


Figure 8. System Flow

Let us make some remarks on the proposed system whose architecture is presented in Figure 7. The keyword extraction is defined as a binary classification where each word is classified into keyword or non-keyword. Each word is encoded into a string vector instead of a numerical vector and is classified directly. Among words which are included in the input text, ones which are classified into keyword are selected. We need to add the text classification module for predicting the domain of the input text automatically.

D. Connection with Text Clustering

This section is concerned with the connection of the keyword extraction with the text clustering. In the previous section, we mentioned the keyword extraction as an instance of domain dependent classification. The domains are automatically constructed from the text collection by clustering texts. The AHC algorithm which is proposed in [3] is used for implementing the text clustering. This section is intended to describe the keyword extraction system which is connected with the text clustering for automating the construction of domains.

The architecture of the text clustering system which was proposed in [3] is illustrated in Figure 9. The texts in the group are encoded into string vectors, and the AHC algorithm which is proposed in [3] is adopted as the approach. It starts with singletons as many as items, computes the similarities of all possible cluster pairs, and merge the cluster pair with its highest similarity into one cluster. The two steps are iterated until the number of clusters reach the desired number. We may consider replacing the AHC algorithm by its variants for improving the speed and the performance.

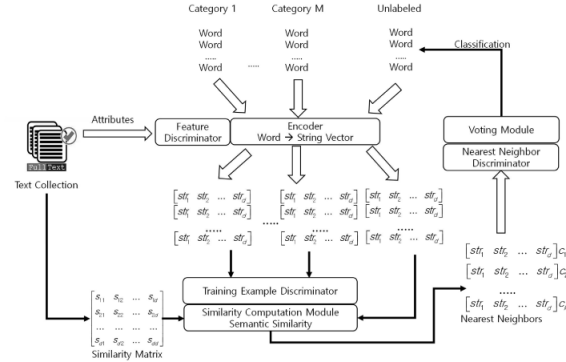


Figure 9. Text Clustering System

The connection of the keyword extraction with the text clustering is illustrated in Figure 10. The M domains are defined by clustering texts in a group, initially and automatically. A novice text is given as the input, and its domain, domain D , is decided by its similarities with domains. A classifier which corresponds to domain D is nominated and classify each word in a novice text into keyword or non-keyword. The M keyword extraction systems are viewed as independent ones, each of which classifies each word into one of the two categories.

The execution flow of keyword extraction which is connected with the text clustering is illustrated in Figure 11. Unlabeled texts are clustered into subgroups, each of which is a domain. Sample words which are labeled with keyword or non-keyword are generated from each domain. These sample words are provided for training the classifier in each keyword extraction system. Each classifier is used for

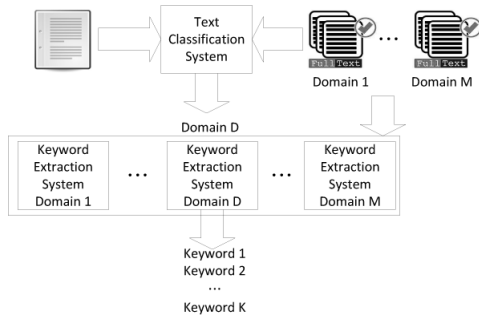


Figure 10. Connection of Keyword Extraction with Text Clustering

judging whether each word which is indexed from the input text is keyword or not.

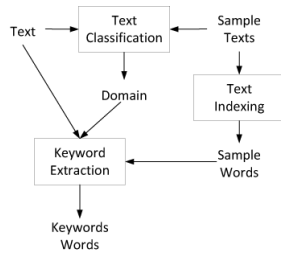


Figure 11. Execution Flow of Keyword Extraction connected with Text Clustering

Let us make some remarks on the connection of the keyword extraction with the text clustering. It is the process of segmenting a text group into subgroups based on their content-based similarities. The role of text clustering is to define the domains initially in the architecture of connecting the keyword extraction with the text clustering. In each domain, sample words are generated, and its corresponding classifier is trained with them. In the future research, we consider using the keyword extraction for clustering texts in the opposite direction.

REFERENCES

- [1] T. Jo, "Automatic Text Summarization using String Vector based K Nearest Neighbor", 6005-6016, Journal of Intelligent and Fuzzy Systems, 35, 2018.
- [2] T. Jo, "Modifying K Nearest Neighbor into String Vector based Version for Extracting Keywords from News Articles", 43-46, The Proceedings of International Conference on Applied Cognitive Computing, 2018.
- [3] T. Jo, "Semantic String Operation for Specializing AHC Algorithm for Text Clustering", 1083-1100, Annals of Mathematics and Artificial Intelligence, 2019.